- Character constant
  - Enclosed in single quotes,
  - forexample: 'z'
- A string is a series of characters treated as a single unit.
  - May include letters, digits and various special characters such as
     +, -, \*, / and \$.
- A string is an array of characters ending with a null character
   ('\0')
- String
  - Enclosed in double quotes,
  - forexample: "I like C++"

- All strings end with null (' \0')
- Character array

```
char color[ 5 ] = "blue";
```

- Creates 5 element char array color
- Null character ('\0') implicitly added
- Alternative forcharacter array

```
char color[ 5 ] = { 'b', 'l', 'u', 'e', '\0' };
```

						color[0]	b	0012FF48
	0	1	2	3	4	color[1]	1	0012FF49
color	b	1	u	e	/0	color[2]	u	0012FF4A
·						color[3]	e	0012FF4B
						color[4]	\0	0012FF4D

- All strings end with null (' \0')
- Character array

```
char color[] = "blue";
```

- Creates 5 element char array color
- Null character ('\0') implicitly added
- Alternative forcharacter array

```
char color[] = \{ 'b', 'l', 'u', 'e', '\setminus 0' \};
```

					color[0]	b	0012FF48
0	1	2	3	4	color[1]	1	0012FF49
b	1	u	e	/0	color[2]	u	0012FF4A
				-	color[3]	e	0012FF4B
					color[4]	\0	0012FF4D
					0 1 2 3 4 b l u e \0	0 1 2 3 4 color[1] b 1 u e \0 color[2] color[3]	b 1 u e \0 color[2] u color[3] e

Input from keyboard

```
char color[ 10 ];
cin >> color;
```

- Puts user input in string
  - Stops at first whitespace character
  - Adds nul 1 character
- If too much text entered, run time error
- Printing strings

```
cout << color << endl;</pre>
```

- Does not work forother array types
- Characters printed until **null** found

```
int main()
{
   char string1[ 20 ];
   char string2[] = "string literal";
   cout << "Enter the string \"hello there\": ";</pre>
   cin >> string1; // reads "hello"
   cout << "string1 is: " << string1 << "\nstring2 is: " << string2;</pre>
   cout << "\nstring1 with spaces between characters is: \n";</pre>
   for( int i = 0; string1[ i ] != ' \setminus 0'; i++ )
      cout << string1[ i ] << ' ';</pre>
   cin >> string1; // reads "there"
   cout << "\nstring1 is: " << string1 << endl;</pre>
```

```
Enter the string "hello there": hello there
string1 is: hello
string2 is: string literal
string1 with spaces between characters is:
h e l l o
string1 is: there
```

	0	1	2	3	4	5	6	7	8	9
s1										

int main()	s1[0]	0012FF48
{     char s1[ 10 ];	s1[1]	0012FF49
	s1[2]	0012FF4A
cin >> s1; // reads "hello"	s1[3]	0012FF4B
cout << "string1 is: " << s1;	s1[4]	0012FF4C
for( int i = 0; s1[ i ] != '\0'; i++ )	s1[5]	0012FF4D
cout << s1[ i ] << ' ';	s1[6]	0012FF4E
cin >> s1; // reads "there"	s1[7]	0012FF4F
cout << "\nstring1 is: " << s1 << endl;	s1[8]	0012FF50
}	s1[9]	0012FF51

```
int main()
                                                     s1[0]
                                                                  0012FF48
                                                                  0012FF49
                                                     s1[1]
   char s1[ 10 ];
                                                     s1[2]
                                                                  0012FF4A
   cin >> s1; // reads "hello"
                                                                  0012FF4B
                                                     s1[3]
                                                                  0012FF4C
   cout << "string1 is: " << s1;</pre>
                                                     s1[4]
                                                     s1[5]
                                                             \0
                                                                  0012FF4D
   for( int i = 0; s1[ i ] != ' \setminus 0'; i++ )
                                                                  0012FF4E
                                                     s1[6]
      cout << s1[ i ] << ' ';
                                                     s1[7]
                                                                  0012FF4F
   cin >> s1; // reads "there"
                                                     s1[8]
                                                                  0012FF50
   cout << "\nstring1 is: " << s1 << endl;</pre>
                                                                  0012FF51
                                                     s1[9]
```

i nt	main()	s1[0]	t	0012FF48
{	char s1[ 10 ];	s1[1]	h	0012FF49
		s1[2]	e	0012FF4A
	cin >> s1; // reads "hello"	s1[3]	r	0012FF4B
	cout << "string1 is: " << s1;	s1[4]	e	0012FF4C
	<pre>for( int i = 0; s1[ i ] != '\0'; i++ )</pre>	s1[5]	\0	0012FF4D
	cout << s1[ i ] << ' ';	s1[6]		0012FF4E
	<pre>cin &gt;&gt; s1; // reads "there" cout &lt;&lt; "\nstring1 is: " &lt;&lt; s1 &lt;&lt; endl;</pre>	s1[7]		0012FF4F
}		s1[8]		0012FF50
		s1[9]		0012FF51

```
s1[0]
                                                                         0012FF48
                                                                     \mathbf{S}
                                                                         0012FF49
                                                            s1[1]
                                                                    t
                                                            s1[2]
                                                                         0012FF4A
                                                                    r
                                                            s1[3]
                                                                         0012FF4B
           2
              3
                  4
                      5
                         6
                             7
                                8
                                    9
                                       10 11 12 13 14
                                                                     i
                                                            s1[4]
                                                                         0012FF4C
s2
       t
              i
                             1
                                i
                                    t
                                                     /0
           r
                  n
                      g
                                       \mathbf{e}
                                           r
                                              a
                                                                    n
                                                                         0012FF4D
                                                            s1[5]
                                                                    g
                                                            s1[6]
                                                                         0012FF4E
     int main()
                                                            s1[7]
                                                                    1
                                                                         0012FF4F
      {
         char s2[] = "string literal";
                                                            s1[8]
                                                                         0012FF50
                                                                    i
                                                            s1[9]
                                                                         0012FF51
                                                                    t
         cout << "string2 is: " << s2;
     }
                                                          s1[10]
                                                                         0012FF52
                                                                     \mathbf{e}
                                                          s1[11]
                                                                         0012FF53
                                                                    r
                                                          s1[12]
                                                                         0012FF54
                                                                     a
                                                          s1[13]
                                                                     1
                                                                         0012FF55
                                                                         0012FF56
                                                          s1[14]
                                                                    \0
```

```
#include <iostream>
#include <cctype>
using namespace std;
voi d convertToUppercase( char * );
int main()
{
   char phrase[] = "characters and $32.98";
   cout << "The phrase before conversion is: " << phrase;</pre>
   convertToUppercase( phrase );
   cout << "\nThe phrase after conversion is: " << phrase << endl;</pre>
voi d convertToUppercase( char *sPtr )
   for( ; *sPtr != '\0'; sPtr++ )
      if( islower( *sPtr ) )
         *sPtr = toupper( *sPtr );
```

The phrase before conversion is: characters and \$32.98 The phrase after conversion is: CHARACTERS AND \$32.98

```
phrase[0]
                                                                 0012FF48
                                                   phrase[1]
                                                               h
                                                                  0012FF49
                                                                  0012FF4A
                                                   phrase[2]
                                                               a
int main()
                                                   phrase[3]
                                                                  0012FF4B
                                                                  0012FF4C
                                                   phrase[4]
                                                               a
   char phrase[] = "characters and $32.98";
                                                   phrase[5]
                                                                  0012FF4D
   convertToUppercase( phrase );
                                                   phrase[6]
                                                                  0012FF4E
                                                               t
                                                                  0012FF4F
                                                   phrase[7]
voi d convertToUppercase( char *sPtr )
                                                                  0012FF50
                                                   phrase[8]
                                                               r
   for( ; *sPtr != '\0'; ++sPtr )
                                                   phrase[9]
                                                                  0012FF51
      if ( islower( *sPtr ) )
                                                                  0012FF52
                                                  phrase[10]
         *sPtr = toupper( *sPtr );
                                                  phrase[11]
                                                                  0012FF53
                                                               a
                                                  phrase[12]
                                                                  0012FF54
                                                               n
                                                                  0012FF55
                                                  phrase[13]
                                                               d
                                                  phrase[14]
                                                                  0012FF56
                                                  phrase[15]
                                                                  0012FF57
                                                                  0012FF58
                                                  phrase[16]
                                                                  0012FF59
                                                  phrase[17]
```

```
phrase[0]
                                                                 0012FF48
                                                                 0012FF49
                                                   phrase[1]
                                 0012FE74
                       0012FF48
                 sPtr
                                                                 0012FF4A
                                                   phrase[2]
                                                              a
int main()
                                                   phrase[3]
                                                                 0012FF4B
                                                                 0012FF4C
                                                   phrase[4]
                                                              a
   char phrase[] = "characters and $32.98";
                                                   phrase[5]
                                                                 0012FF4D
   convertToUppercase( phrase );
                                                   phrase[6]
                                                                 0012FF4E
                                                              t
                                                                 0012FF4F
                                                   phrase[7]
voi d convertToUppercase( char *sPtr )
                                                                 0012FF50
                                                   phrase[8]
                                                              r
   for( ; *sPtr != '\0'; ++sPtr )
                                                   phrase[9]
                                                                 0012FF51
      if ( islower( *sPtr ) )
                                                  phrase[10]
                                                                 0012FF52
         *sPtr = toupper( *sPtr );
                                                  phrase[11]
                                                                 0012FF53
                                                              a
                                                  phrase[12]
                                                                 0012FF54
                                                              n
                                                                 0012FF55
                                                  phrase[13]
                                                              d
                                                  phrase[14]
                                                                 0012FF56
                                                  phrase[15]
                                                                 0012FF57
                                                                 0012FF58
                                                  phrase[16]
                                                                 0012FF59
                                                  phrase[17]
```

```
phrase[0]
                                                                 0012FF48
                                                                 0012FF49
                                                   phrase[1]
                                 0012FE74
                       0012FF48
                 sPtr
                                                                 0012FF4A
                                                   phrase[2]
                                                              a
int main()
                                                   phrase[3]
                                                                 0012FF4B
                                                                 0012FF4C
                                                   phrase[4]
                                                              a
   char phrase[] = "characters and $32.98";
                                                   phrase[5]
                                                                 0012FF4D
   convertToUppercase( phrase );
                                                   phrase[6]
                                                                 0012FF4E
                                                              t
                                                                 0012FF4F
                                                   phrase[7]
voi d convertToUppercase( char *sPtr )
                                                                 0012FF50
                                                   phrase[8]
                                                              r
   for( ; *sPtr != '\0'; ++sPtr )
                                                   phrase[9]
                                                                 0012FF51
      if ( islower( *sPtr ) )
                                                  phrase[10]
                                                                 0012FF52
         *sPtr = toupper( *sPtr );
                                                  phrase[11]
                                                                 0012FF53
                                                              a
                                                  phrase[12]
                                                                 0012FF54
                                                              n
                                                                 0012FF55
                                                  phrase[13]
                                                              d
                                                  phrase[14]
                                                                 0012FF56
                                                  phrase[15]
                                                                 0012FF57
                                                                 0012FF58
                                                  phrase[16]
                                                                 0012FF59
                                                  phrase[17]
```

```
phrase[0]
                                                                 0012FF48
                                                                 0012FF49
                                                   phrase[1]
                       0012FF49
                                 0012FE74
                 sPtr
                                                                 0012FF4A
                                                   phrase[2]
                                                               a
int main()
                                                   phrase[3]
                                                                 0012FF4B
                                                                 0012FF4C
                                                   phrase[4]
                                                               a
   char phrase[] = "characters and $32.98";
                                                   phrase[5]
                                                                 0012FF4D
   convertToUppercase( phrase );
                                                   phrase[6]
                                                                 0012FF4E
                                                               t
                                                                 0012FF4F
                                                   phrase[7]
voi d convertToUppercase( char *sPtr )
                                                                 0012FF50
                                                   phrase[8]
                                                               r
   for( ; *sPtr != '\0'; ++sPtr )
                                                   phrase[9]
                                                                 0012FF51
      if ( islower( *sPtr ) )
                                                  phrase[10]
                                                                  0012FF52
         *sPtr = toupper( *sPtr );
                                                  phrase[11]
                                                                 0012FF53
                                                               a
                                                  phrase[12]
                                                                 0012FF54
                                                               n
                                                                 0012FF55
                                                  phrase[13]
                                                               d
                                                  phrase[14]
                                                                  0012FF56
                                                  phrase[15]
                                                                 0012FF57
                                                                 0012FF58
                                                  phrase[16]
                                                                 0012FF59
                                                  phrase[17]
```

```
phrase[0]
                                                                 0012FF48
                                                                 0012FF49
                                                   phrase[1]
                       0012FF49
                                 0012FE74
                 sPtr
                                                                 0012FF4A
                                                   phrase[2]
                                                               a
int main()
                                                   phrase[3]
                                                                 0012FF4B
                                                                 0012FF4C
                                                   phrase[4]
                                                               a
   char phrase[] = "characters and $32.98";
                                                   phrase[5]
                                                                 0012FF4D
   convertToUppercase( phrase );
                                                   phrase[6]
                                                                 0012FF4E
                                                               t
                                                                 0012FF4F
                                                   phrase[7]
voi d convertToUppercase( char *sPtr )
                                                                 0012FF50
                                                   phrase[8]
                                                               r
   for( ; *sPtr != '\0'; ++sPtr )
                                                   phrase[9]
                                                                 0012FF51
      if ( islower( *sPtr ) )
                                                  phrase[10]
                                                                  0012FF52
         *sPtr = toupper( *sPtr );
                                                  phrase[11]
                                                                 0012FF53
                                                               a
                                                  phrase[12]
                                                                 0012FF54
                                                               n
                                                                 0012FF55
                                                  phrase[13]
                                                               d
                                                  phrase[14]
                                                                  0012FF56
                                                  phrase[15]
                                                                 0012FF57
                                                                 0012FF58
                                                  phrase[16]
                                                                 0012FF59
                                                  phrase[17]
```

```
phrase[0]
                                                                 0012FF48
                                                                 0012FF49
                                                   phrase[1]
                       0012FF4A
                                 0012FE74
                 sPtr
                                                                 0012FF4A
                                                   phrase[2]
                                                              a
int main()
                                                   phrase[3]
                                                                 0012FF4B
                                                                 0012FF4C
                                                   phrase[4]
                                                              a
   char phrase[] = "characters and $32.98";
                                                   phrase[5]
                                                                 0012FF4D
   convertToUppercase( phrase );
                                                   phrase[6]
                                                                 0012FF4E
                                                              t
                                                                 0012FF4F
                                                   phrase[7]
voi d convertToUppercase( char *sPtr )
                                                                 0012FF50
                                                   phrase[8]
                                                              r
   for( ; *sPtr != '\0'; ++sPtr )
                                                   phrase[9]
                                                                 0012FF51
      if ( islower( *sPtr ) )
                                                  phrase[10]
                                                                 0012FF52
         *sPtr = toupper( *sPtr );
                                                  phrase[11]
                                                                 0012FF53
                                                              a
                                                  phrase[12]
                                                                 0012FF54
                                                              n
                                                                 0012FF55
                                                  phrase[13]
                                                              d
                                                  phrase[14]
                                                                 0012FF56
                                                  phrase[15]
                                                                 0012FF57
                                                                 0012FF58
                                                  phrase[16]
                                                                 0012FF59
                                                  phrase[17]
```

```
phrase[0]
                                                                 0012FF48
                                                                 0012FF49
                                                   phrase[1]
                       0012FF4A
                                 0012FE74
                 sPtr
                                                                 0012FF4A
                                                   phrase[2]
                                                              a
int main()
                                                   phrase[3]
                                                                 0012FF4B
                                                                 0012FF4C
                                                   phrase[4]
                                                              a
   char phrase[] = "characters and $32.98";
                                                   phrase[5]
                                                                 0012FF4D
   convertToUppercase( phrase );
                                                   phrase[6]
                                                                 0012FF4E
                                                              t
                                                                 0012FF4F
                                                   phrase[7]
voi d convertToUppercase( char *sPtr )
                                                                 0012FF50
                                                   phrase[8]
                                                              r
   for( ; *sPtr != '\0'; ++sPtr )
                                                   phrase[9]
                                                                 0012FF51
      if ( islower( *sPtr ) )
                                                  phrase[10]
                                                                 0012FF52
         *sPtr = toupper( *sPtr );
                                                  phrase[11]
                                                                 0012FF53
                                                              a
                                                  phrase[12]
                                                                 0012FF54
                                                              n
                                                                 0012FF55
                                                  phrase[13]
                                                              d
                                                  phrase[14]
                                                                 0012FF56
                                                  phrase[15]
                                                                 0012FF57
                                                                 0012FF58
                                                  phrase[16]
                                                                 0012FF59
                                                  phrase[17]
```

```
phrase[0]
                                                                 0012FF48
                                                                 0012FF49
                                                   phrase[1]
                       0012FF4B
                                 0012FE74
                 sPtr
                                                                 0012FF4A
                                                   phrase[2]
                                                              a
int main()
                                                   phrase 3
                                                                 0012FF4B
                                                                 0012FF4C
                                                   phrase[4]
                                                              a
   char phrase[] = "characters and $32.98";
                                                   phrase[5]
                                                                 0012FF4D
   convertToUppercase( phrase );
                                                   phrase[6]
                                                                 0012FF4E
                                                              t
                                                                 0012FF4F
                                                   phrase[7]
voi d convertToUppercase( char *sPtr )
                                                                 0012FF50
                                                   phrase[8]
                                                              r
   for( ; *sPtr != '\0'; ++sPtr )
                                                   phrase[9]
                                                                 0012FF51
      if ( islower( *sPtr ) )
                                                  phrase[10]
                                                                 0012FF52
         *sPtr = toupper( *sPtr );
                                                  phrase[11]
                                                                 0012FF53
                                                              a
                                                  phrase[12]
                                                                 0012FF54
                                                              n
                                                                 0012FF55
                                                  phrase[13]
                                                              d
                                                  phrase[14]
                                                                 0012FF56
                                                  phrase[15]
                                                                 0012FF57
                                                                 0012FF58
                                                  phrase[16]
                                                                 0012FF59
                                                  phrase[17]
```

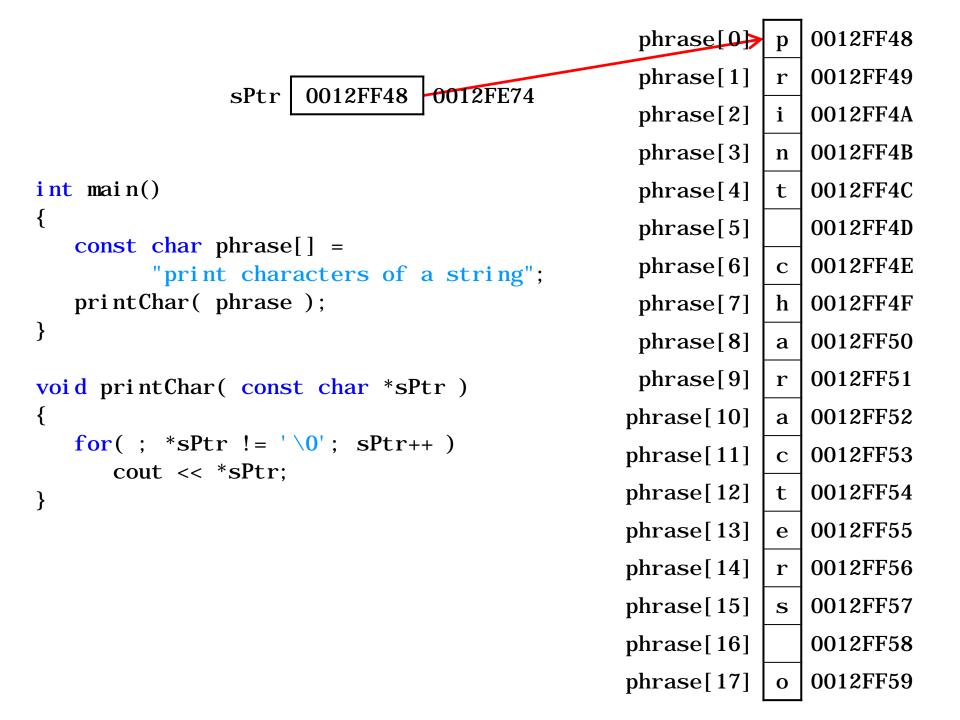
```
int main()
{
   char phrase[] = "characters and $32.98";
   cout << "The phrase before conversion is: " << phrase;</pre>
   convertToUppercase( phrase );
   cout << "\nThe phrase after conversion is: " << phrase << endl;</pre>
int main()
{
   string phrase( "characters and $32.98");
   cout << "The phrase before conversion is: " << phrase;</pre>
   convertToUppercase( phrase );
   cout << "\nThe phrase after conversion is: " << phrase << endl;</pre>
```

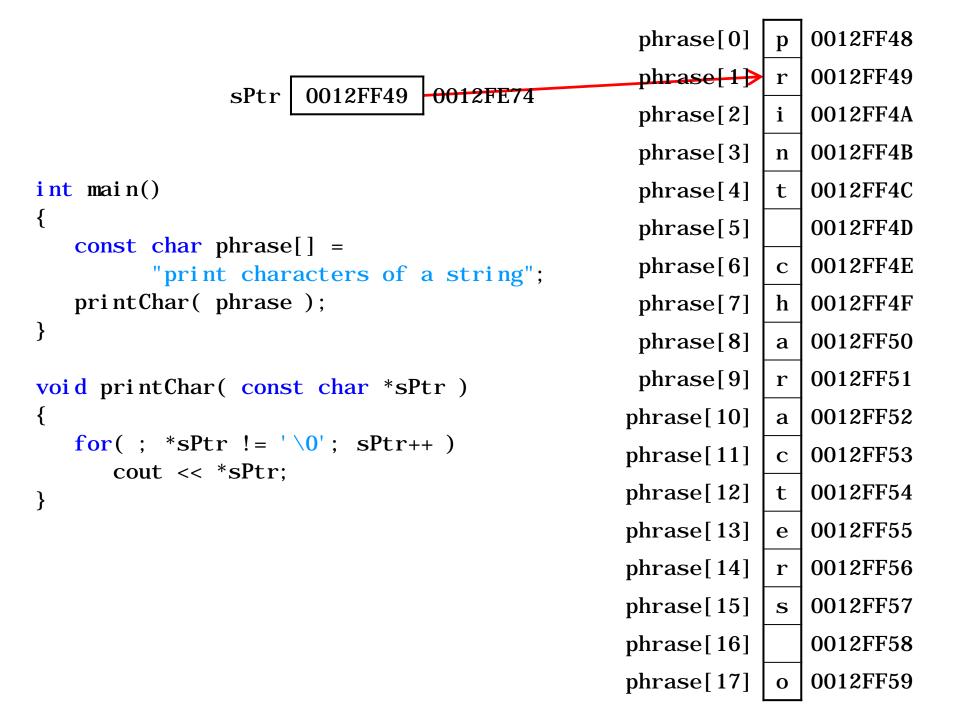
```
voi d convertToUppercase( char *s )
   for( int i = 0; i < strlen(s); i++)
     if( islower( s[i] ) )
        s[i] = toupper(s[i]);
void convertToUppercase( string s )
   for( int i = 0; i < s.length(); i++)
      if( islower( s[i] ) )
         s[i] = toupper(s[i]);
```

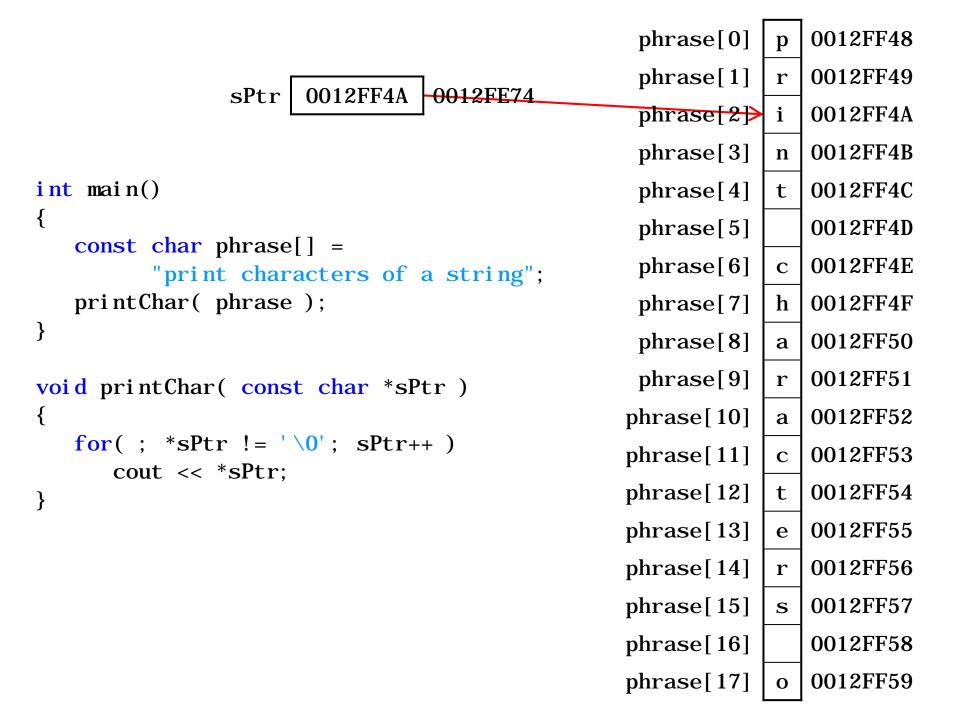
```
#include <iostream>
using namespace std;
void printCharacters( const char * );
int main()
{
   const char phrase[] = "print characters of a string";
   cout << "The string is: \n";</pre>
   printCharacters( phrase );
   cout << endl;</pre>
}
void printCharacters( const char *sPtr )
{
   for( ; *sPtr != '\0'; sPtr++ )
      cout << *sPtr;</pre>
}
```

The string is: print characters of a string

```
phrase[0]
                                                                0012FF48
                                                  phrase[1]
                                                                0012FF49
                                                              r
                                                                0012FF4A
                                                  phrase[2]
                                                  phrase[3]
                                                                0012FF4B
                                                              n
int main()
                                                                0012FF4C
                                                  phrase[4]
                                                              t
                                                  phrase[5]
                                                                 0012FF4D
   const char phrase[] =
                                                  phrase[6]
                                                                0012FF4E
         "print characters of a string";
   printChar( phrase );
                                                                0012FF4F
                                                  phrase[7]
                                                              h
                                                                0012FF50
                                                  phrase[8]
                                                              a
                                                  phrase[9]
                                                                0012FF51
                                                              r
voi d printChar( const char *sPtr )
                                                 phrase[10]
                                                                0012FF52
                                                              a
   for( ; *sPtr != '\0'; sPtr++ )
                                                                0012FF53
                                                 phrase[11]
      cout << *sPtr:
                                                 phrase[12]
                                                                0012FF54
                                                              t
                                                                0012FF55
                                                 phrase[13]
                                                 phrase[14]
                                                                0012FF56
                                                              r
                                                 phrase[15]
                                                                0012FF57
                                                              S
                                                                 0012FF58
                                                 phrase[16]
                                                                0012FF59
                                                 phrase[17]
```







```
phrase[0]
                                                                0012FF48
                                                  phrase[1]
                                                                0012FF49
                                                             r
                      0012FF4B
                                0012FE74
                sPtr
                                                                0012FF4A
                                                  phrase[2]
                                                  phrase 3
                                                                0012FF4B
                                                             n
int main()
                                                                0012FF4C
                                                  phrase[4]
                                                             t
                                                  phrase[5]
                                                                0012FF4D
   const char phrase[] =
                                                  phrase[6]
                                                                0012FF4E
         "print characters of a string";
   printChar( phrase );
                                                                0012FF4F
                                                  phrase[7]
                                                                0012FF50
                                                  phrase[8]
                                                             a
                                                  phrase[9]
                                                                0012FF51
                                                             r
voi d printChar( const char *sPtr )
                                                 phrase[10]
                                                                0012FF52
                                                             a
   for( ; *sPtr != '\0'; sPtr++ )
                                                                0012FF53
                                                 phrase[11]
      cout << *sPtr:
                                                 phrase[12]
                                                                0012FF54
                                                             t
                                                                0012FF55
                                                 phrase[13]
                                                 phrase[14]
                                                                0012FF56
                                                             r
                                                 phrase[15]
                                                                0012FF57
                                                             S
                                                                0012FF58
                                                 phrase[16]
                                                                0012FF59
                                                 phrase[17]
```

```
int main()
{
   const char phrase[] = "print characters of a string";
   printCharacters( phrase );
void printCharacters( const char s[] )
   for( unsigned int i = 0; i < strlen(s); i++)
      cout << s[i];
int main()
{
   const string phrase( "print characters of a string" );
   printCharacters( phrase );
void printCharacters( const string s )
   for (unsigned int i = 0; i < s.length(); i ++ )
      cout << s[i];
```

```
#include <iostream>
using namespace std;
voi d copy1( char *, const char * );
voi d copy2( char *, const char * );
int main()
   char string1[ 10 ];
   char *string2 = "Hello";
   char string3[ 10 ];
   char string4[] = "Good Bye";
   copy1( string1, string2 );
   cout << "string1 = " << string1 << endl;</pre>
   copy2( string3, string4 );
   cout << "string3 = " << string3 << endl;</pre>
}
```

```
void copy1( char * s1, const char * s2 )
{
   for( int i = 0; ( s1[ i ] = s2[ i ] ) != '\0'; i++ )
    ;
}

void copy2( char *s1, const char *s2 )
{
   for( ; ( *s1 = *s2 ) != '\0'; s1++, s2++ )
    ;
}
```

```
void copy1( char * s1, const char * s2 )
{
    for( int i = 0; ( s1[ i ] = s2[ i ] ) != '\0'; i++ )
      ;
}

void copy2( char *s1, const char *s2 )
{
    for( ; ( *s1 = *s2 ) != '\0'; s1++, s2++ )
      ;
}
```

```
string1 = Hello
string3 = Good Bye
```

```
int main()
{
   char string1[ 10 ];
   char *string2 = "Hello";
   copy1( string1, string2 );
   cout << "string1 = " << string1 << endl;</pre>
}
voi d copy1( char * s1, const char * s2 )
{
   for( int i = 0; ( s1[i] = s2[i] ) != '\0'; i++ )
}
```

```
copy1( string1, string2 );
 voi d copy1( char *s1, const char *s2 )
    for( int i = 0; ( s1[i] = s2[i] ) != '\0'; i++)
 }
                                              string1[0]
                                                                  0012FF54
                                              string1[1]
                                                                  0012FF55
                                                                  0012FF56
                                              string1[2]
                                              string1[3]
                                                                  0012FF57
                                              string1[4]
                                                                  0012FF58
                                              string1[5]
                                                                  0012FF59
                                              string2[0]
                                                                  0041782C
                                                             H
                                              string2[1]
                                                                  0041782D
                                                             \mathbf{e}
                                                             1
                                              string2[2]
                                                                  0041782E
                    0012FF48
         0041782C
string2
                                                             1
                                                                  0041782F
                                              string2[3]
                                              string2[4]
                                                                  00417831
                                                             \mathbf{o}
                                              string2[5]
                                                            /0
                                                                  00417832
```

```
copy1( string1, string2 );
 voi d copy1( char *s1, const char *s2 )
    for( int i = 0; ( s1[ i ] = s2[ i ] ) != ' \setminus 0'; i++ )
                                      s1[0]
                                               string1[0]
                                                                   0012FF54
 }
                                               string1[1]
                                      s1[1]
                                                                   0012FF55
                                      s1[2]
                                                                   0012FF56
                                               string1[2]
                    0012FE48
     s1
         0012FF54
                                      s1[3]
                                               string1[3]
                                                                   0012FF57
     s2
         0041782C
                    0012FE4C
                                      s1[4]
                                                                   0012FF58
                                               string1[4]
                                      s1[5]
                                               string1[5]
                                                                   0012FF59
                                               string2[0]
                                                              H
                                                                   0041782C
                                      s2[0]
                                                                   0041782D
                                      s2[1]
                                               string2[1]
                                                              \mathbf{e}
                                                              1
                                      s2[2]
                                                                   0041782E
                                               string2[2]
                     0012FF48
string2
         0041782C
                                                              1
                                      s2[3]
                                               string2[3]
                                                                   0041782F
                                      s2[4]
                                               string2[4]
                                                                   00417831
                                                              0
                                      s2[5]
                                               string2[5]
                                                             /0
                                                                   00417832
```

```
int main()
{
   char string3[ 10 ];
   char string4[] = "Good Bye";
   copy2( string3, string4 );
   cout << "string3 = " << string3 << endl;</pre>
}
voi d copy2( char *s1, const char *s2 )
{
  for(; (*s1 = *s2) != '\0'; s1++, s2++)
}
```

```
for(; (*s1 = *s2) != '\0'; s1++, s2++)
                                           string3[0]
                                                              0012FF34
                                *(s1-2)
                                *(s1-1)
                                           string3[1]
                                                              0012FF35
                                           string3[2]
                                                              0012FF36
                                     *s1
                                *(s1+1)
                                           string3[3]
                                                              0012FF37
                                *(s1+2)
                                                              0012FF38
                                           string3[4]
                                           string3[5]
                                                              0012FF39
                                *(s1+3)
                                           string4[0]
                                                              0012FF20
                                *(s2-2)
                                                         G
                                *(s2-1)
                                           string4[1]
                                                              0012FF21
                                                         0
                                                              0012FF22
                                    *s2
                                           string4[2]
                                                         0
                                *(s2+1)
                                           string4[3]
                                                         d
                                                              0012FF23
                                *(s2+2)
                                           string4[4]
                                                         B
                                                              0012FF24
                                *(s2+3)
                                           string4[5]
                                                              0012FF25
                                                         y
                                                              0012FF26
                                *(s2+4)
                                           string4[6]
                                                         \mathbf{e}
                                                         \0
                                *(s2+5)
                                           string4[7]
                                                              0012FF27
                                *(s2+6)
                                           string4[8]
                                                              0012FF28
```

```
for(; (*s1 = *s2) != '\0'; s1++, s2++)
                                            string3[0]
                                                              0012FF34
                                *(s1-2)
                                            string3[1]
                                *(s1-1)
                                                              0012FF35
                                     *s1
                                            string3[2]
                                                              0012FF36
                                *(s1+1)
                                                              0012FF37
                                            string3[3]
      0012FF34
                0012FE48
                                                              0012FF38
  s1
                                *(s1+2)
                                            string3[4]
                0012FE4C
                                                              0012FF39
                                *(s1+3)
                                            string3[5]
  s2
      0012FF20
                                            string4[0]
                                                              0012FF20
                                *(s2-2)
                                                          G
                                *(s2-1)
                                            string4[1]
                                                              0012FF21
                                                          0
                                                              0012FF22
                                     *s2
                                            string4[2]
                                                          0
                                *(s2+1)
                                                          d
                                                              0012FF23
                                            string4[3]
                                *(s2+2)
                                            string4[4]
                                                          B
                                                              0012FF24
                                *(s2+3)
                                            string4[5]
                                                              0012FF25
                                                          y
                                                              0012FF26
                                *(s2+4)
                                            string4[6]
                                                          \mathbf{e}
                                *(s2+5)
                                            string4[7]
                                                         \0
                                                              0012FF27
                                *(s2+6)
                                            string4[8]
                                                              0012FF28
```

```
for(; (*s1 = *s2) != '\0'; s1++, s2++)
                                                              0012FF34
                                           string3[0]
                                *(s1-2)
                                                         G
                                *(s1-1)
                                           string3[1]
                                                              0012FF35
                                           string3[2]
                                                              0012FF36
                                     *s1
                                *(s1+1)
                                           string3[3]
                                                              0012FF37
                                *(s1+2)
      0012FF35
                0012FE48
                                                              0012FF38
  s1
                                           string3[4]
                                                              0012FF39
                0012FE4C
                                *(s1+3)
                                           string3[5]
  s2
      0012FF21
                                *(s2-2)
                                                              0012FF20
                                          string4[0]
                                                         G
                                           string4[1]
                                *(s2-1)
                                                              0012FF21
                                                         0
                                                              0012FF22
                                     *s2
                                           string4[2]
                                                         0
                                *(s2+1)
                                                         d
                                                              0012FF23
                                           string4[3]
                                *(s2+2)
                                           string4[4]
                                                         B
                                                              0012FF24
                                *(s2+3)
                                           string4[5]
                                                              0012FF25
                                                         y
                                                              0012FF26
                                *(s2+4)
                                           string4[6]
                                                         \mathbf{e}
                                *(s2+5)
                                           string4[7]
                                                         \0
                                                              0012FF27
                                *(s2+6)
                                           string4[8]
                                                              0012FF28
```

```
for(; ( *s1 = *s2 ) != '\0'; s1++, s2++ )
                                                               0012FF34
                                            string3[0]
                                *(s1-2)
                                                          G
                                *(s1-1)
                                            string3[1]
                                                               0012FF35
                                                          0
                                            string3[2]
                                                               0012FF36
                                     *s1
                                            string3[3]
                                                               0012FF37
                                *(s1+1)
                                *(s1+2)
      0012FF36
                0012FE48
                                                               0012FF38
  s1
                                            string3[4]
                                                               0012FF39
                0012FE4C
                                *(s1+3)
                                            string3[5]
  s2
      0012FF22
                                                               0012FF20
                                *(52-2)
                                            string4[0]
                                                          G
                                *(s2-1)
                                            string4[1]
                                                               0012FF21
                                                          0
                                            string4[2]
                                                               0012FF22
                                     *s2
                                                          0
                                *(s2+1)
                                            string4[3]
                                                          d
                                                               0012FF23
                                                          B
                                *(s2+2)
                                            string4[4]
                                                               0012FF24
                                *(s2+3)
                                            string4[5]
                                                               0012FF25
                                                          y
                                                               0012FF26
                                *(s2+4)
                                            string4[6]
                                                          \mathbf{e}
                                                          \0
                                *(s2+5)
                                            string4[7]
                                                               0012FF27
                                *(s2+6)
                                            string4[8]
                                                               0012FF28
```

```
char *strcpy( char *s1, const char *s2 );
                 Copies the string s2 into the character array s1. The value of s1 is
                 returned.
char *strncpy( char *s1, const char *s2, size_t n );
                 Copies at most n characters of the string s2 into the character array s1.
                 The value of s1 is returned.
char *strcat( char *s1, const char *s2 );
                 Appends the string s2 to s1. The first character of s2 overwrites the
                 terminating null character of s1. The value of s1 is returned.
char *strncat( char *s1, const char *s2, size_t n );
                 Appends at most n characters of string s2 to string s1. The first
                 character of s2 overwrites the terminating null character of s1. The
                 value of s1 is returned.
```

int strcmp( const char \*s1, const char \*s2 );

Compares the string s1 with the string s2. The function returns a value of zero, less than zero (usually -1) or greater than zero (usually 1) if s1 is equal to, less than or greater than s2, respectively.

int strncmp( const char \*s1, const char \*s2, size\_t n );

Compares up to n characters of the string s1 with the string s2. The function returns zero, less than zero or greater than zero if the n-character portion of s1 is equal to, less than or greater than the corresponding n-character portion of s2, respectively.

char \*strtok( char \*s1, const char \*s2 );

A sequence of calls to strtok breaks string s1 into "tokens" — logical pieces such as words in a line of text. The string is broken up based on the characters contained in string s2. forinstance, if we were to break the string "this: is: a: string" into tokens based on the character ': ', the resulting tokens would be "this", "is", "a" and "string". Function strtok returns only one token at a time, however. The first call contains s1 as the first argument, and subsequent calls to continue tokenizing the same string contain NULL as the first argument. A pointer to the current token is returned by each call. If there are no more tokens when the function is called, NULL is returned.

size\_t strlen( const char \*s );

Determines the length of string s. The number of characters preceding the terminating null character is returned.

- Copying strings
  - char \*strcpy( char \*s1, const char \*s2 )
    - Copies second argument into first argument
      - First argument must be large enough to store string and terminating null character
  - char \*strncpy( char \*s1, const char \*s2, size\_t n )
    - Specifies number of characters to be copied from string into array
    - Does not necessarily copy terminating null character

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
   char x[] = "Happy Birthday to You";
   char y[ 25 ];
   char z[ 15 ];
   strcpy( y, x );
   cout << "The string in array x is: " << x
      << "\nThe string in array y is: " << y << '\n';</pre>
   strncpy(z, x, 14); // copy first 14 characters of x into z
   z[14] = ' \cdot 0'; // append '\0' to z's contents
   cout << "The string in array z is: " << z << endl;</pre>
```

			ı		•
	y[0]	0012FF4C	<b>x</b> [0]	н	0012FF68
	y[1]	0012FF4D	x[1]	a	0012FF69
	y[2]	0012FF4E	x[2]	p	0012FF6A
	y[3]	0012FF4F	<b>x</b> [3]	p	0012FF6B
	y[4]	0012FF50	x[4]	Y	0012FF6C
strcpy(y, x);	y[5]	0012FF51	<b>x</b> [5]		0012FF6D
	y[6]	0012FF52	<b>x</b> [6]	В	0012FF6E
	y[7]	0012FF53	<b>x</b> [7]	i	0012FF6F
	y[8]	0012FF54	<b>x</b> [8]	r	0012FF70
	y[9]	0012FF55	<b>x</b> [9]	t	0012FF71
	y[10]	0012FF56	<b>x</b> [10]	h	0012FF72
	y[11]	0012FF57	<b>x</b> [11]	đ	0012FF73
	y[12]	0012FF58	x[12]	a	0012FF74
	y[13]	0012FF59	x[13]	Y	0012FF75
	y[14]	0012FF5A	x[14]		0012FF76
	y[15]	0012FF5B	<b>x</b> [15]	t	0012FF77
	y[16]	0012FF5C	<b>x</b> [16]	0	0012FF78
	y[17]	0012FF5D	<b>x</b> [17]		0012FF79
	y[18]	0012FF5E	x[18]	Y	0012FF7A
	y[19]	0012FF5F	x[19]	0	0012FF7B
	y[20]	0012FF60	x[20]	u	0012FF7C
	y[21]	0012FF61	x[21]	/0	0012FF7D

```
int main()
{
   char x[] = "Happy Birthday to You"; // string length 21
   char y[ 25 ];
   char z[ 15 ];
   strcpy(y, x); // copy contents of x into y
   cout << "The string in array x is: " << x
      << "\nThe string in array y is: " << y << '\n';</pre>
   // copy first 14 characters of x into z
   strncpy(z, x, 14); // does not copy null character
   z[14] = ' \cdot 0'; // append ' \cdot 0' to z's contents
   cout << "The string in array z is: " << z << endl;</pre>
```

```
int main()
{
   char x[] = "Happy Birthday to You"; // string length 21
   char y[ 25 ];
   char z[ 15 ];
   strcpy_s(y, 25, x) // copy contents of x into y
   cout << "The string in array x is: " << x
      << "\nThe string in array y is: " << y << '\n';</pre>
   // copy first 14 characters of x into z
   strncpy_s(z, 15, x, 14); // places '\0' after last character
   cout << "The string in array z is: " << z << endl;</pre>
```

```
int main()
{
   string x( "Happy Birthday to You" ); // string length 21
   string y;
   string z;
  y = x; // copy contents of x into y
   cout << "The string in array x is: " << x
      << "\nThe string in array y is: " << y << '\n';</pre>
   // copy first 14 characters of x into z
   z = x. substr(0, 14);
   cout << "The string in array z is: " << z << endl;</pre>
```

The string in array x is: Happy Birthday to You

The string in array y is: Happy Birthday to You

The string in array z is: Happy Birthday

- Concatenating strings
  - char \*strcat( char \*s1, const char \*s2 )
    - Appends second argument to first argument
    - First character of second argument replaces null character terminating first argument
    - Ensure first argument large enough to store concatenated result and null character
  - char \*strncat( char \*s1, const char \*s2, size\_t n )
    - Appends specified number of characters from second argument to first argument
    - Appends terminating null character to result

```
int main()
{
   char s1[20] = "Happy";
   char s2[] = "New Year ";
   char s3[ 40 ] = "";
   cout << "s1 = " << s1 << " \ns2 = " << s2;
   strcat( s1, s2 );
   cout << "\nAfter strcat(s1, s2): \ns1 = " << s1 << "\ns2 = " << s2;
   // concatenate first 6 characters of s1 to s3
   strncat(s3, s1, 6); // places '\0' after last character
   cout << "\n\nAfter strncat(s3, s1, 6):\ns1 = " << s1
        << "\ns3 = " << s3:
   strcat( s3, s1);
   cout << "\n\nAfter strcat(s3, s1):\ns1 = " << s1</pre>
      << "\ns3 = " << s3 << endl:
```

```
s1 = Happy
s2 = New Year
After strcat(s1, s2):
s1 = Happy New Year
s2 = New Year
After strncat(s3, s1, 6):
s1 = Happy New Year
s3 = Happy
After streat(s3, s1):
s1 = Happy New Year
s3 = Happy Happy New Year
```

			_			_
			0012FF58	<b>s1</b> [0]	Н	0012FF6C
			0012FF59	s1[1]	a	0012FF6D
			0012FF5A	s1[2]	р	0012FF6E
			0012FF5B	<b>s1</b> [3]	р	0012FF6F
strcat(s1, s2);			0012FF5C	s1[4]	У	0012FF70
			0012FF5D	<b>s</b> 1[5]		0012FF71
			0012FF5E	<b>s1</b> [6]	/0	0012FF72
			0012FF5F	s1[7]		0012FF73
	<b>s2[0]</b>	N	0012FF60	s1[8]		0012FF74
	s2[1]	е	0012FF61	<b>s1</b> [9]		0012FF75
	s2[2]	w	0012FF62	<b>s1</b> [10]		0012FF76
	<b>s2[3]</b>		0012FF63	s1[11]		0012FF77
	s2[4]	Y	0012FF64	s1[12]		0012FF78
	<b>s2</b> [5]	е	0012FF65	<b>s</b> 1[13]		0012FF79
	<b>s2[6]</b>	a	0012FF66	s1[14]		0012FF7A
	s2[7]	r	0012FF67	s1[15]		0012FF7B
	s2[8]		0012FF68	s1[16]		0012FF7C
	s2[9]	/0	0012FF69	s1[17]		0012FF7D
			0012FF6A	s1[18]		0012FF7E
			0012FF6B	s1[19]		0012FF7F

```
int main()
{
  char s1[20] = "Happy";
   char s2[] = "New Year ";
   char s3[ 40 ] = "";
  cout << "s1 = " << s1 << "\ns2 = " << s2;
  strcat(s1, s2); // concatenate s2 to s1 (length 15)
  cout << s1 << "\ns2 = " << s2;
  strncat(s3, s1, 6); // places '\0' after last character
  cout << s1 << "\ns3 = " << s3;
   strcat(s3, s1); // concatenate s1 to s3
   cout << s1 << "\ns3 = " << s3 << endl;
```

```
int main()
{
   char s1[20] = "Happy";
   char s2[] = "New Year ";
   char s3[ 40 ] = "";
  cout << "s1 = " << s1 << "\ns2 = " << s2;
  strcat_s(s1, 20, s2); // concatenate s2 to s1 (length 15)
  cout << s1 << "\ns2 = " << s2;
  strncat_s(s3, 40, s1, 6); // places '\0' after last character
  cout << s1 << "\ns3 = " << s3;
   strcat_s(s3, 40, s1); // concatenate s1 to s3
   cout << s1 << "\ns3 = " << s3 << endl;
```

```
int main()
{
   string s1( "Happy " );
   string s2( "New Year " );
   string s3;
  cout << "s1 = " << s1 << "\ns2 = " << s2;
   s1 += s2; // concatenate s2 to s1 (length 15)
   cout << s1 << "\ns2 = " << s2;
   s3 += s1. substr(0, 6);
   cout << s1 << "\ns3 = " << s3;
   s3 += s1; // concatenate s1 to s3
   cout << s1 << "\ns3 = " << s3 << endl;
```

- Comparing strings
  - Characters represented as numeric codes
    - Strings compared using numeric codes
  - ASCII character sets
    - "American Standard Code forInformation Interchage"

- Comparing strings
  - int strcmp( const char \*s1, const char \*s2 )
    - Compares character by character
    - Returns
      - Zero if strings equal
      - Negative value if first string less than second string
      - Positive value if first string greater than second string
  - - Compares up to specified number of characters
    - Stops comparing if reaches null character in one of arguments

```
#include <iostream>
#include <i omanip>
#include <cstring>
using namespace std;
int main()
{
   char *s1 = "Happy New Year";
   char *s2 = "Happy New Year";
   char *s3 = "Happy Holidays";
   cout << "s1 = " << s1 << "\ns2 = " << s2 << "\ns3 = " << s3
      << "\n\nstrcmp(s1, s2) = " << setw( 2 ) << strcmp( s1, s2 )
      << "\nstrcmp(s1, s3) = " << setw( 2 ) << strcmp( s1, s3 )
      << "\nstrcmp(s3, s1) = " << setw( 2 ) << strcmp( s3, s1 );
   cout << "\n\nstrncmp(s1, s3, 6) = " << setw(2)
      << strncmp(s1, s3, 6) << "\nstrncmp(s1, s3, 7) = " << setw(2)
      << strncmp(s1, s3, 7) << "\nstrncmp(s3, s1, 7) = " << setw(2)
      << strncmp( s3, s1, 7 ) << endl;</pre>
```

```
s1 = Happy New Year
s2 = Happy New Year
s3 = Happy Holidays

strcmp(s1, s2) = 0
strcmp(s1, s3) = 1
strcmp(s3, s1) = -1

strncmp(s1, s3, 6) = 0
strncmp(s1, s3, 7) = 1
strncmp(s3, s1, 7) = -1
```

- Determining string lengths
  - size\_t strlen( const char \*s )
    - Returns number of characters in string
      - Terminating null character not included in length

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
   char *string1 = "abcdefghijklmnopqrstuvwxyz";
   char *string2 = "four";
   char *string3 = "Boston";
   cout << "The length of \"" << string1 << "\" is " << strlen( string1 )</pre>
      << "\nThe length of \"" << string2 << "\" is " << strlen( string2 )</pre>
      << "\nThe length of \"" << string3 << "\" is " << strlen( string3 )
      << endl;
```

The length of "abcdefghijklmnopqrstuvwxyz" is 26 The length of "four" is 4 The length of "Boston" is 6

```
int main()
   char *string1 = "abcdefghijklmnopqrstuvwxyz";
   char *string2 = "four";
   char *string3 = "Boston";
   cout << "The length of \"" << string1 << "\" " << strlen( string1 )</pre>
      << "\nThe length of \"" << string2 << "\" " << strlen( string2 )</pre>
      << "\nThe length of \"" << string3 << "\" " << strlen( string3 );
}
int main()
   string string1( "abcdefghijklmnopqrstuvwxyz" );
   string string2( "four" );
   string string3( "Boston" );
   cout << "The length of \"" << string1 << "\" " << string1.length()</pre>
      << "\nThe length of \"" << string2 << "\" " << string2.length()</pre>
      << "\nThe length of \"" << string3 << "\" " << string3.length();</pre>
```

#### Tokenizing

- Breaking strings into tokens, separated by delimiting characters
- Tokens usually logical units, such as words (separated by spaces)
- "This is my string" has 4 word tokens (separated by spaces)
- char \*strtok( char \*s1, const char \*s2 )
  - Multiple calls required
    - First call contains two arguments, string to be tokenized and string containing delimiting characters
      - Finds next delimiting character and replaces with null character
    - Subsequent calls continue tokenizing
      - Call with first argument NULL

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
      char string[] = "Programming is learned by writing programs";
      cout << string << endl << endl;</pre>
      char *token = strtok( string, " " ); // get the first token
      while( token != nullptr )
         cout << token << endl:
         token = strtok( nullptr, " " ); // get next token
      cout << endl;
      cout << string << endl << endl;</pre>
```

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
      char string[] = "Programming is learned by writing programs";
      cout << string << endl << endl;</pre>
      char *nextToken = nullptr;
      char *token = strtok_s( string, " ", &nextToken );
      while( token != nullptr )
         cout << token << endl;
         token = strtok_s( nullptr, " ", &nextToken );
      cout << endl;
      cout << string << endl << endl;</pre>
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                              ptr
                                                             a
                                                             r
                                                             \mathbf{e}
                                           nextToken
                                                             y
                                                             \mathbf{o}
              Output
                                                             u
                                                             \0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             Η
                                                             \mathbf{O}
                                                             W
                                                                        string
                                               ptr
                                                             a
                                                             r
                                                             \mathbf{e}
                                           nextToken
                                                                         str
                                                             y
                                                             0
              Output
                                                             u
                                                             \0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                           Η
                                                           0
                                                           W
                                                                     string
                                             ptr
                                                           a
                                                           r
                                                           \mathbf{e}
                                          nextToken
                                                                       str
                                                           y
                                                           0
             Output
                                                           u
                                                           \0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             Η
                                                             \mathbf{O}
                                                             W
                                                                        string
                                               ptr
                                                             a
                                                             r
                                                             \mathbf{e}
                                           nextToken
                                                                          str
                                                             y
                                                             0
              Output
                                                             u
                                                             \0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             \mathbf{O}
                                                             W
                                                                        string
                                               ptr
                                                             /0
                                                             a
                                                             r
                                                             \mathbf{e}
                                           nextToken
                                                                          str
                                                             y
                                                             0
              Output
                                                             u
                                                             \0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             Η
                                                             \mathbf{O}
                                                             W
                                                                        string
                                               ptr
                                                             \0
                                                             a
                                                             r
                                                             \mathbf{e}
                                           nextToken
                                                                          str
                                                             y
                                                             0
              Output
                                                             u
                                                             \0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             \mathbf{O}
                                                             W
                                                                        string
                                               ptr
                                                             /0
                                                             a
                                                             r
                                                             \mathbf{e}
                                           nextToken
                                                                          str
                                                             y
                                                             O
              Output
                                                             u
                                                             /0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                           H
                                                           O
                                                           W
                                                           \0
                                             ptr
                                                           a
                                                           r
                                                           \mathbf{e}
                                          nextToken
                                                           y
                                                           0
             Output
                                                           u
                                                           \0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                           H
                                                           O
                                                           W
                                                           \0
                                             ptr
                                                           a
       How
                                                           r
                                                           \mathbf{e}
                                          nextToken
                                                           y
                                                           0
             Output
                                                           u
                                                           \0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                           H
                                                           O
                                                           W
                                                           \0
                                                                     string
                                             ptr
                                                           a
       How
                                                           r
                                                           \mathbf{e}
                                          nextToken
                                                                       str
                                                           y
                                                           O
             Output
                                                           u
                                                           /0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                           H
                                                           O
                                                           W
                                                           \0
                                                                     string
                                             ptr
                                                           a
       How
                                                           r
                                                           \mathbf{e}
                                          nextToken
                                                                       str
                                                           y
                                                           O
             Output
                                                           u
                                                           /0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                           H
                                                           O
                                                           W
                                                           \0
                                                                     string
                                             ptr
                                                           a
       How
                                                           r
                                                           \mathbf{e}
                                          nextToken
                                                                       str
                                                           y
                                                           O
             Output
                                                           u
                                                           /0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                           H
                                                           O
                                                           W
                                                           /0
                                                                     string
                                             ptr
                                                           a
       How
                                                           r
                                                           \mathbf{e}
                                          nextToken
                                                                       str
                                                           y
                                                           O
             Output
                                                           u
                                                           \0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                                             /0
                                                                        string
                                               ptr
                                                             a
       How
                                                             r
                                                             \mathbf{e}
                                                             \0
                                           nextToken
                                                                          str
                                                             y
                                                             \mathbf{o}
              Output
                                                             u
                                                             /0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                                             /0
                                                                        string
                                               ptr
                                                             a
       How
                                                             r
                                                             \mathbf{e}
                                                             \0
                                           nextToken
                                                                          str
                                                             y
                                                             \mathbf{O}
              Output
                                                             u
                                                             /0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                                             /0
                                                                        string
                                               ptr
                                                             a
       How
                                                             r
                                                             \mathbf{e}
                                                             \0
                                           nextToken
                                                                          str
                                                             y
                                                             \mathbf{O}
              Output
                                                             u
                                                             /0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                               ptr
                                                             \0
                                                             a
       How
                                                             r
                                                             \mathbf{e}
                                                             \0
                                           nextToken
                                                             y
                                                             \mathbf{O}
              Output
                                                             u
                                                             \0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                               ptr
                                                             \0
                                                             a
        How
                                                             r
                                                             \mathbf{e}
        are
                                                             \0
                                           nextToken
                                                             y
                                                             \mathbf{O}
              Output
                                                             u
                                                             \0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                                                        string
                                               ptr
                                                             /0
                                                             a
       How
                                                             r
                                                             \mathbf{e}
       are
                                                             \0
                                           nextToken
                                                                          str
                                                             y
                                                             \mathbf{o}
              Output
                                                             u
                                                             /0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                                                        string
                                               ptr
                                                             /0
                                                             a
       How
                                                             r
                                                             \mathbf{e}
       are
                                                             \0
                                           nextToken
                                                                          str
                                                             y
                                                             \mathbf{o}
              Output
                                                             u
                                                             /0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                                             /0
                                                                        string
                                               ptr
                                                             a
        How
                                                             r
                                                             \mathbf{e}
        are
                                                             \0
                                           nextToken
                                                                          str
                                                             y
                                                             \mathbf{o}
              Output
                                                             u
                                                             /0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                                             /0
                                                                        string
                                               ptr
                                                             a
       How
                                                             r
                                                             \mathbf{e}
       are
                                                             \0
                                           nextToken
                                                                          str
                                                             y
                                                             \mathbf{o}
              Output
                                                             u
                                                             \0
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                                             /0
                                                                        string
                                               ptr
                                                             a
       How
                                                             r
                                                             \mathbf{e}
       are
                                                             \0
                                           nextToken
                                                                          str
                                                             y
                                                             \mathbf{o}
              Output
                                                             u
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                               ptr
                                                             \0
                                                             a
       How
                                                             r
                                                             \mathbf{e}
       are
                                                             \0
                                           nextToken
                                                             y
                                                             \mathbf{o}
              Output
                                                             u
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                               ptr
                                                             \0
                                                             a
       How
                                                             r
                                                             \mathbf{e}
       are
                                                             \0
       you
                                           nextToken
                                                             y
                                                             \mathbf{o}
              Output
                                                             u
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                                             \0
                                                                        string
                                               ptr
                                                             a
       How
                                                             r
                                                             \mathbf{e}
       are
                                                             \0
       you
                                           nextToken
                                                                          str
                                                             y
                                                             \mathbf{o}
              Output
                                                             u
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                                             \0
                                                                        string
                                               ptr
                                                             a
       How
                                                             r
                                                             \mathbf{e}
       are
                                                             \0
       you
                                           nextToken
                                                                          str
                                                             y
                                                             \mathbf{o}
              Output
                                                             u
```

```
int main()
   char string[] = "How are you";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, " ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
      ptr = strtok_s( nullptr, " ", &nextToken );
                                                             H
                                                             O
                                                             W
                                               ptr
                                                             \0
                                                             a
       How
                                                             r
                                                             \mathbf{e}
       are
                                                             \0
       you
                                           nextToken
                                                             y
                                                             \mathbf{o}
              Output
                                                             u
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                              H
      ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                              \mathbf{e}
                                                              1
                                                              1
                                                ptr
                                                              0
                                                              W
                                                              \mathbf{o}
                                            nextToken
                                                              r
                                                              d
               Output
                                                              /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                            H
      ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                            \mathbf{e}
                                                            1
                                                            1
                                                                      string
                                              ptr
                                                            0
                                                            W
                                                            0
                                           nextToken
                                                                        str
                                                            r
                                                            d
              Output
                                                            /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                            H
      ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                            \mathbf{e}
                                                            1
                                                            1
                                                                      string
                                              ptr
                                                            0
                                                            W
                                                            0
                                           nextToken
                                                                        str
                                                            r
                                                            d
              Output
                                                            /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                            H
      ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                            \mathbf{e}
                                                            1
                                                            1
                                                                      string
                                              ptr
                                                            0
                                                            W
                                                            0
                                           nextToken
                                                                        str
                                                            r
                                                            d
              Output
                                                            /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                            H
      ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                            \mathbf{e}
                                                            1
                                                            1
                                                                      string
                                              ptr
                                                            0
                                                            W
                                                            0
                                           nextToken
                                                                        str
                                                            r
                                                            d
              Output
                                                            /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
       cout << ptr << '\n';
                                                                H
       ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                                \mathbf{e}
                                                                1
                                                                1
                                                                           string
                                                 ptr
                                                                \mathbf{o}
                                                                W
                                                                \mathbf{o}
                                              nextToken
                                                                             str
                                                                r
                                                                d
               Output
                                                                /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
       cout << ptr << '\n';
                                                                H
       ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                                \mathbf{e}
                                                                1
                                                                1
                                                                           string
                                                 ptr
                                                                \mathbf{o}
                                                                \0
                                                                W
                                                                \mathbf{o}
                                              nextToken
                                                                             str
                                                                r
                                                                d
               Output
                                                                /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
       cout << ptr << '\n';
                                                                H
       ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                                \mathbf{e}
                                                                1
                                                                1
                                                                           string
                                                 ptr
                                                                \mathbf{o}
                                                                \0
                                                                W
                                                                \mathbf{o}
                                              nextToken
                                                                             str
                                                                r
                                                                d
               Output
                                                                /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
       cout << ptr << '\n';
                                                                H
       ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                                \mathbf{e}
                                                                1
                                                                1
                                                                           string
                                                 ptr
                                                                \mathbf{o}
                                                                \0
                                                                W
                                                                \mathbf{o}
                                              nextToken
                                                                             str
                                                                r
                                                                d
               Output
                                                                /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
       cout << ptr << '\n';
                                                                H
       ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                                \mathbf{e}
                                                                1
                                                                1
                                                                           string
                                                 ptr
                                                                \mathbf{o}
                                                                \0
                                                                W
                                                                \mathbf{o}
                                              nextToken
                                                                             str
                                                                r
                                                                d
               Output
                                                                /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
       cout << ptr << '\n';
                                                                H
       ptr = strtok_s( nullptr, "\",! ", &nextToken )/,
                                                                \mathbf{e}
                                                 ptr
                                                                \mathbf{o}
                                                                \0
                                                                W
                                                                \mathbf{o}
                                              nextToken
                                                                r
                                                                d
               Output
                                                                /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                              H
      ptr = strtok_s( nullptr, "\",! ", &nextToken )/,
                                                              \mathbf{e}
                                                ptr
                                                              0
                                                              \0
                                                              W
         Hello
                                                              \mathbf{o}
                                            nextToken
                                                              r
                                                              d
               Output
                                                              /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                            H
      ptr = strtok_s( nullptr, "\",! ", &nextToken )/,
                                                            \mathbf{e}
                                                                       string
                                              ptr
                                                            0
                                                            \0
                                                            W
        Hello
                                                            0
                                           nextToken
                                                                        str
                                                            r
                                                            d
              Output
                                                            /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                              H
      ptr = strtok_s( nullptr, "\",! ", &nextToken )/,
                                                              \mathbf{e}
                                                                         string
                                                ptr
                                                              0
                                                              \0
                                                              W
         Hello
                                                              \mathbf{o}
                                            nextToken
                                                                           str
                                                              r
                                                              d
               Output
                                                              /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                              H
      ptr = strtok_s( nullptr, "\",! ", &nextToken )/,
                                                              \mathbf{e}
                                                                         string
                                                ptr
                                                              0
                                                              \0
                                                              W
         Hello
                                                              \mathbf{o}
                                            nextToken
                                                                           str
                                                              r
                                                              d
               Output
                                                              /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
       cout << ptr << '\n';
                                                                H
       ptr = strtok_s( nullptr, "\",! ", &nextToken )/,
                                                                \mathbf{e}
                                                                           string
                                                 ptr
                                                                0
                                                                \0
                                                                \mathbf{W}
         Hello
                                                                \mathbf{o}
                                              nextToken
                                                                             str
                                                                r
                                                                d
               Output
                                                                /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
       cout << ptr << '\n';
                                                                  H
       ptr = strtok_s( nullptr, "\",! ", &nextToken )/,
                                                                  \mathbf{e}
                                                                             string
                                                   ptr
                                                                  \mathbf{O}
                                                                  \0
                                                                  \mathbf{W}
         Hello
                                                                  \mathbf{o}
                                               nextToken
                                                                               str
                                                                  r
                                                                  d
                Output
                                                                  /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
       cout << ptr << '\n';
                                                                  H
       ptr = strtok_s( nullptr, "\",! ", &nextToken )/,
                                                                  \mathbf{e}
                                                                              string
                                                   ptr
                                                                  \mathbf{O}
                                                                  \0
                                                                  \mathbf{W}
         Hello
                                                                  \mathbf{o}
                                               nextToken
                                                                                str
                                                                  r
                                                                  d
                                                                  /0
                Output
                                                                   11
                                                                  /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
       cout << ptr << '\n';
                                                                  H
       ptr = strtok_s( nullptr, "\",! ", &nextToken )/,
                                                                  \mathbf{e}
                                                                             string
                                                   ptr
                                                                  \mathbf{O}
                                                                  \0
                                                                  \mathbf{W}
         Hello
                                                                  \mathbf{o}
                                               nextToken
                                                                               str
                                                                  r
                                                                  d
                                                                  /0
                Output
                                                                  /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                              H
      ptr = strtok_s( nullptr, "\",! ", &nextToken )/,
                                                              \mathbf{e}
                                                ptr
                                                                         string
                                                              0
                                                              \0
                                                              W
         Hello
                                                              \mathbf{o}
                                            nextToken
                                                                           str
                                                              r
                                                              d
                                                              \0
               Output
                                                              /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                                H
      ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                                \mathbf{e}
                                                                1
                                                                1
                                                 ptr
                                                                \mathbf{o}
                                                               \0
                                                                W
         Hello
                                                                \mathbf{o}
                                              nextToken
                                                                r
                                                                d
                                                                \0
               Output
                                                                \0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
       cout << ptr << '\n';
                                                                H
       ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                                \mathbf{e}
                                                                1
                                                                1
                                                 ptr
                                                                \mathbf{o}
                                                                \0
                                                                W
         Hello
                                                                \mathbf{o}
         worl d
                                              nextToken
                                                                r
                                                                d
                                                                \0
               Output
                                                                \0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                              H
      ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                              \mathbf{e}
                                                              1
                                                              1
                                                                         string
                                                ptr
                                                              0
                                                              \0
                                                              W
         Hello
                                                              \mathbf{o}
         worl d
                                            nextToken
                                                                           str
                                                              r
                                                              d
                                                              \0
               Output
                                                              /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                              H
      ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                              \mathbf{e}
                                                              1
                                                              1
                                                                         string
                                                ptr
                                                              0
                                                              \0
                                                              W
         Hello
                                                              \mathbf{o}
         worl d
                                            nextToken
                                                                           str
                                                              r
                                                              d
                                                              \0
               Output
                                                              /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                              H
      ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                              \mathbf{e}
                                                              1
                                                              1
                                                                         string
                                                ptr
                                                              0
                                                              \0
                                                              W
         Hello
                                                              \mathbf{o}
         worl d
                                            nextToken
                                                                           str
                                                              r
                                                              d
                                                              \0
               Output
                                                              \0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                              H
      ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                              \mathbf{e}
                                                              1
                                                              1
                                                                         string
                                                ptr
                                                              0
                                                              \0
                                                              W
         Hello
                                                              \mathbf{o}
         worl d
                                                                          str
                                            nextToken
                                                              r
                                                              d
                                                              \0
               Output
                                                              /0
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                              H
      ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                              \mathbf{e}
                                                              1
                                                              1
                                                                         string
                                                ptr
                                                              0
                                                              \0
                                                              W
         Hello
                                                              \mathbf{o}
         worl d
                                            nextToken
                                                                          str
                                                              r
                                                              d
                                                              \0
               Output
```

```
int main()
   char string[] = "\"Hello, world!\"";
   char *nextToken;
   char *ptr;
   ptr = strtok_s( string, "\",! ", &nextToken );
   while( ptr != nullptr )
      cout << ptr << '\n';
                                                                H
      ptr = strtok_s( nullptr, "\",! ", &nextToken );
                                                                \mathbf{e}
                                                                1
                                                                1
                                                 ptr
                                                                O
                                                               /0
                                                                W
         Hello
                                                                \mathbf{o}
         worl d
                                             nextToken
                                                                r
                                                                d
                                                               \0
               Output
                                                               \sqrt{0}
```

```
char *strtok_s( char *string, const char *control, char **context )
{
   char *it = ( string == nullptr ) ? *context : string;
   // Find the next nondelimiter
  while( belong( *it, control ) )
      it++;
   char * const tokenFirst = it;
   // Find the next delimiter
   for( ; *it != '\0'; it++ )
      if( belong( *it, control ) )
      {
         *(it++) = ' \setminus 0';
         break;
   *context = it:
   return ( tokenFirst == it ) ? nullptr : tokenFirst;
```

```
char *strtok_s( char *string, const char *control, char **context )
   char *it = ( string == nullptr ) ? *context : string;
   // Find the next nondelimiter
                                                         H
   while( belong( *it, control ) )
      i t++:
                                                         \mathbf{O}
                                                         W
   char * const tokenFirst = it;
                                                                  string
   // Find the next delimiter
                                        ptr |
   for( ; *it != '\0'; it++ )
                                                         a
      if( belong( *it, control ) )
                                                         r
         *( it++ ) = '\0';
                                                         \mathbf{e}
                                  nextToken
                                                                       i t
         break;
                                                         y
                                                         0
                                    context
   *context = it:
                                                         u
                                                        \0
   return ( tokenFirst == it ) ? nullptr : tokenFirst;
}
              ptr = strtok_s( string, " ", &nextToken );
              while( ptr != nullptr )
                 cout << ptr << '\n';
                 ptr = strtok_s( nullptr, " ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *it;
                                                           H
  if( string == nullptr )
    it = *context:
                                                           O
  el se
                                                           W
    it = string;
                                                                    string
  // Find the next nondelimiter
                                          ptr
                                                           a
  while( belong( *it, control ) )
                                                           r
    i t++;
                                                           \mathbf{e}
  string = it;
                                    nextToken
  // Find the next delimiter
                                                           y
  for(; *it != '\0'; it++ )
                                                           0
    if( belong( *it, control ) )
                                                           u
        *(it++) = ' \setminus 0';
                                            context
                                                           /0
        break:
  *context = it:
                         ptr = strtok_s( string, " ", &nextToken );
                         while( ptr != nullptr )
  if( string == it )
    return nullptr;
                            cout << ptr << '\n';
  el se
                             ptr = strtok_s( nullptr, " ", &nextToken );
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
  if( string == NULL )
    str = *context:
  el se
    str = string;
  // Find the next delimiter
  while( belong( *str, control ) )
    str++;
  string = str;
  // Find the next nondelimiter
  for( ; *str != '\0'; str++ )
    if( belong( *str, control ) )
        *(str++) = ' \setminus 0';
        break;
  *context = str;
  if( string == str )
    return NULL;
  el se
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                        Н
  if( string == NULL )
    str = *context;
                                                        W
  el se
                                                                   string
                                           ptr
    str = string;
                                                        a
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                         r
    str++:
                                                         e
  string = str;
                                 nextToken
  // Find the next delimiter
                                                                    str
                                                        У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                        u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                           ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  el se
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                        Η
  if( string == NULL )
    str = *context;
                                                        W
  el se
                                                                   string
                                           ptr
    str = string;
                                                        a
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                        r
    str++;
                                                        e
  string = str;
                                nextToken
  // Find the next delimiter
                                                                    str
                                                        У
  for( ; *str != '\0'; str++ )
                                                        0
    if( belong( *str, control ) )
                                                        u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                           ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  else
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                        Η
  if( string == NULL )
    str = *context;
                                                        W
  el se
                                                                   string
                                           ptr
    str = string;
                                                        a
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                         r
    str++;
                                                        e
  string = str;
                                 nextToken
  // Find the next delimiter
                                                                    str
                                                        У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                        u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                           ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  el se
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                        Η
  if( string == NULL )
    str = *context;
                                                        W
  el se
                                                                   string
                                           ptr
    str = string;
                                                        a
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                        r
    str++;
                                                        e
  string = str;
                                nextToken
  // Find the next delimiter
                                                                    str
                                                        У
  for( ; *str != '\0'; str++ )
                                                        0
    if( belong( *str, control ) )
                                                        u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                           ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  else
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Η
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                                                   string
                                                        \ 0
                                           ptr
    str = string;
                                                         a
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                         r
    str++;
                                                         e
  string = str;
                                 nextToken
  // Find the next delimiter
                                                                    str
                                                         У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  else
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Η
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                                                   string
                                                        \ 0
                                           ptr
    str = string;
                                                         a
  // Find the next nondelimiter
                                                         r
  while( belong( *str, control ) )
    str++;
                                                         e
  string = str;
                                 nextToken
  // Find the next delimiter
                                                                    str
                                                         У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  else
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Η
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                                                   string
                                                        \ 0
                                           ptr
    str = string;
                                                         a
  // Find the next nondelimiter
                                                         r
  while( belong( *str, control ) )
    str++:
                                                         e
  string = str;
                                 nextToken
  // Find the next delimiter
                                                                    str
                                                         У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  else
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Н
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                           ptr
                                                        \ 0
    str = string;
                                                         a
  // Find the next nondelimiter
                                                         r
  while( belong( *str, control ) )
    str++:
                                                         e
  string = str;
                                 nextToken
  // Find the next delimiter
                                                         У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
  el se
                               ptr = strtok_s( NULL, " ", &nextToken );
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Н
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                                                   string
                                                        \ 0
                                           ptr
    str = string;
                                                         a
  // Find the next nondelimiter
                                                         r
  while( belong( *str, control ) )
    str++:
                                                         e
  string = str;
                                 nextToken
  // Find the next delimiter
                                                                    str
                                                         У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  el se
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Н
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                           ptr
                                                        \ 0
                                                                   string
    str = string;
                                                         a
  // Find the next nondelimiter
                                                         r
  while( belong( *str, control ) )
    str++:
                                                         e
  string = str;
                                 nextToken
  // Find the next delimiter
                                                                    str
                                                         У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  el se
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Н
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                           ptr
                                                        \ 0
                                                                   string
    str = string;
                                                         a
  // Find the next nondelimiter
                                                         r
  while( belong( *str, control ) )
    str++:
                                                         e
  string = str;
                                 nextToken
  // Find the next delimiter
                                                                    str
                                                         У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  else
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Н
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                           ptr
                                                        \ 0
                                                                   string
    str = string;
                                                         a
  // Find the next nondelimiter
                                                         r
  while( belong( *str, control ) )
    str++:
                                                         e
  string = str;
                                 nextToken
  // Find the next delimiter
                                                                    str
                                                         У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  el se
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Н
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                           ptr
                                                         \ 0
                                                                   string
    str = string;
                                                         a
  // Find the next nondelimiter
                                                         r
  while( belong( *str, control ) )
    str++:
                                                         e
  string = str;
                                 nextToken
                                                         \ 0
  // Find the next delimiter
                                                                     str
                                                         У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                            while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  el se
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                        Н
  if( string == NULL )
    str = *context;
                                                        W
  el se
                                          ptr
                                                                  string
                                                        \ 0
    str = string;
                                                        a
  // Find the next nondelimiter
                                                        r
  while( belong( *str, control ) )
    str++:
                                                        e
  string = str;
                                nextToken
                                                        \0
  // Find the next delimiter
                                                                    str
                                                        У
  for( ; *str != '\0'; str++ )
                                                        0
    if( belong( *str, control ) )
                                                        u
    {
                                        context
       *(str++) = ' \ 0';
       break;
                           ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                              cout << ptr << '\n';
                              ptr = strtok_s( NULL, " ", &nextToken );
  el se
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                        Н
  if( string == NULL )
    str = *context;
                                                        W
  el se
                                          ptr
                                                        \ 0
                                                                  string
    str = string;
                                                        a
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                        r
    str++:
                                                        e
  string = str;
                                nextToken
                                                        \0
  // Find the next delimiter
                                                        У
                                                                    str
  for( ; *str != '\0'; str++ )
                                                        0
    if( belong( *str, control ) )
                                                        u
    {
                                        context
       *(str++) = ' \ 0';
       break;
                           ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                              cout << ptr << '\n';
  el se
                              ptr = strtok_s( NULL, " ", &nextToken );
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Н
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                                        \ 0
                                           ptr
    str = string;
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                         r
    str++:
                                                         e
  string = str;
                                 nextToken
                                                        \ 0
  // Find the next delimiter
                                                         У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
  el se
                               ptr = strtok_s( NULL, " ", &nextToken );
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                        Н
  if( string == NULL )
    str = *context;
                                                        W
  el se
                                                        \ 0
                                                                  string
                                          ptr
    str = string;
                                                        a
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                        r
    str++:
                                                        e
  string = str;
                                nextToken
                                                        \ 0
  // Find the next delimiter
                                                        У
                                                                    str
  for( ; *str != '\0'; str++ )
                                                        0
    if( belong( *str, control ) )
                                                        u
    {
                                        context
        *(str++) = ' \ 0';
       break;
                           ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                              cout << ptr << '\n';
  el se
                              ptr = strtok_s( NULL, " ", &nextToken );
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                        Н
  if( string == NULL )
    str = *context;
                                                        W
  el se
                                                                  string
                                                        \ 0
                                          ptr
    str = string;
                                                        a
  // Find the next nondelimiter
                                                        r
  while( belong( *str, control ) )
    str++:
                                                        e
  string = str;
                                nextToken
                                                        \ 0
  // Find the next delimiter
                                                        У
                                                                    str
  for( ; *str != '\0'; str++ )
                                                        0
    if( belong( *str, control ) )
                                                        u
    {
                                        context
        *(str++) = ' \ 0';
       break;
                           ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                              cout << ptr << '\n';
  el se
                              ptr = strtok_s( NULL, " ", &nextToken );
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Н
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                                                   string
                                                        \ 0
                                           ptr
    str = string;
                                                         a
  // Find the next nondelimiter
                                                         r
  while( belong( *str, control ) )
    str++:
                                                         e
  string = str;
                                 nextToken
                                                        \0
  // Find the next delimiter
                                                         У
                                                                     str
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                            while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  el se
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Н
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                                                   string
                                                        \ 0
                                           ptr
    str = string;
                                                         a
  // Find the next nondelimiter
                                                         r
  while( belong( *str, control ) )
    str++:
                                                         e
  string = str;
                                 nextToken
                                                        \0
  // Find the next delimiter
                                                         У
                                                                     str
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                            while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  el se
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                          Н
  if( string == NULL )
    str = *context;
                                                          W
  el se
                                                                    string
                                                         \ 0
                                            ptr
    str = string;
                                                          a
  // Find the next nondelimiter
                                                          r
  while( belong( *str, control ) )
    str++:
                                                          e
  string = str;
                                 nextToken
                                                         \0
  // Find the next delimiter
                                                          У
                                                                      str
  for( ; *str != '\0'; str++ )
                                                          0
    if( belong( *str, control ) )
                                                          \mathbf{u}
    {
                                         context
        *(str++) = ' \setminus 0';
        break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                            while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  el se
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Н
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                                        \ 0
                                           ptr
    str = string;
                                                         a
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                         r
    str++:
                                                         e
  string = str;
                                 nextToken
                                                        \0
  // Find the next delimiter
                                                         У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                           while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
  el se
                               ptr = strtok_s( NULL, " ", &nextToken );
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Н
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                                        \ 0
                                                                   string
                                           ptr
    str = string;
                                                         a
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                         r
    str++:
                                                         e
  string = str;
                                 nextToken
                                                         \0
  // Find the next delimiter
                                                                     str
                                                         У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
                                         context
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                            while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  el se
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                          Н
  if( string == NULL )
    str = *context;
                                                          W
  el se
                                                         \ 0
                                                                    string
                                           ptr
    str = string;
                                                          a
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                          r
    str++:
                                                          e
  string = str;
                                 nextToken
                                                         \0
  // Find the next delimiter
                                                                      str
                                                          У
  for( ; *str != '\0'; str++ )
                                                          0
    if( belong( *str, control ) )
                                                          \mathbf{u}
    {
                                         context
        *(str++) = ' \setminus 0';
        break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                            while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  else
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                          Н
  if( string == NULL )
    str = *context;
                                                          W
  el se
                                                                    string
                                                         \ 0
                                           ptr
    str = string;
                                                          a
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                          r
    str++:
                                                          e
  string = str;
                                 nextToken
                                                         \0
  // Find the next delimiter
                                                          У
  for( ; *str != '\0'; str++ )
                                                          0
    if( belong( *str, control ) )
                                                          \mathbf{u}
    {
                                         context
        *(str++) = ' \setminus 0';
        break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                            while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
                               ptr = strtok_s( NULL, " ", &nextToken );
  el se
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                         Н
  if( string == NULL )
    str = *context;
                                                         W
  el se
                                                        \ 0
                                           ptr
    str = string;
                                                         a
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                         r
    str++:
                                                         e
  string = str;
                                 nextToken
                                                        \ 0
  // Find the next delimiter
                                                         У
  for( ; *str != '\0'; str++ )
                                                         0
    if( belong( *str, control ) )
                                                         u
    {
        *(str++) = ' \setminus 0';
       break;
                            ptr = strtok_s( string, " ", &nextToken );
  *context = str;
                            while( ptr != NULL )
  if( string == str )
    return NULL;
                               cout << ptr << '\n';
  el se
                               ptr = strtok_s( NULL, " ", &nextToken );
    return string;
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                           Η
  if( string == NULL )
                                                           e
    str = *context;
                                                           1
  el se
                                                           1
    str = string;
                                              ptr
                                                           0
  // Find the next nondelimiter
  while( belong( *str, control ) )
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                           0
  for( ; *str != '\0'; str++ )
                                                           r
    if( belong( *str, control ) )
                                                           1
    {
                                                           d
        *(str++) = ' \setminus 0';
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                            Н
  if( string == NULL )
                                                            e
    str = *context;
                                                            1
  el se
                                                            1
    str = string;
                                                                      string
                                              ptr
                                                            0
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                            •
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                            d
        *(str++) = ' \setminus 0';
                                            context
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                            Η
  if( string == NULL )
                                                            e
    str = *context;
                                                            1
  el se
                                                            1
    str = string;
                                                                      string
                                              ptr
                                                            0
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                            •
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                            d
        *(str++) = ' \setminus 0';
                                            context
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                           Н
  if( string == NULL )
                                                           e
    str = *context;
                                                            1
  el se
                                                            1
    str = string;
                                                                      string
                                              ptr
                                                           0
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                            •
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                           d
        *(str++) = ' \setminus 0';
                                           context
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                            Η
  if( string == NULL )
                                                            e
    str = *context;
                                                            1
  el se
                                                            1
    str = string;
                                                                      string
                                              ptr
                                                            0
  // Find the next nondelimiter
  while( belong( *str, control ) )
                                                            •
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                            d
        *(str++) = ' \setminus 0';
                                            context
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                           Η
  if( string == NULL )
                                                            e
    str = *context;
                                                            1
  el se
                                                            1
    str = string;
                                                                      string
                                              ptr
                                                           0
  // Find the next nondelimiter
  while( belong( *str, control ) )
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                           d
        *(str++) = ' \setminus 0';
                                           context
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                            Η
  if( string == NULL )
                                                            e
    str = *context;
                                                            1
  el se
                                                            1
    str = string;
                                                                      string
                                              ptr
                                                            0
  // Find the next nondelimiter
                                                           \0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                            d
        *(str++) = ' \setminus 0';
                                            context
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                            Η
  if( string == NULL )
                                                            e
    str = *context;
                                                            1
  el se
                                                            1
    str = string;
                                                                      string
                                              ptr
                                                            0
  // Find the next nondelimiter
                                                           \0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                            d
        *(str++) = ' \setminus 0';
                                            context
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                           Η
  if( string == NULL )
                                                            e
    str = *context;
  el se
    str = string;
                                              ptr
                                                           0
  // Find the next nondelimiter
                                                           \ 0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                           d
        *(str++) = ' \setminus 0';
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                           Η
  if( string == NULL )
                                                            e
    str = *context;
  el se
    str = string;
                                                                      string
                                              ptr
                                                           0
  // Find the next nondelimiter
                                                           \ 0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                           d
        *(str++) = ' \setminus 0';
                                           context
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                           Η
  if( string == NULL )
                                                           e
    str = *context;
  el se
    str = string;
                                                                      string
                                              ptr
                                                           0
  // Find the next nondelimiter
                                                           \0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                           0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                           r
    if( belong( *str, control ) )
                                                           1
    {
                                                           d
        *(str++) = ' \setminus 0';
                                           context
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                           Η
  if( string == NULL )
                                                            e
    str = *context;
  el se
    str = string;
                                              ptr
                                                                      string
                                                           0
  // Find the next nondelimiter
                                                           \ 0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                           d
        *(str++) = ' \setminus 0';
                                           context
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                           Η
  if( string == NULL )
                                                            e
    str = *context;
  el se
    str = string;
                                                                      string
                                              ptr
                                                           0
  // Find the next nondelimiter
                                                           \ 0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                           d
        *(str++) = ' \setminus 0';
                                           context
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                           Η
  if( string == NULL )
                                                            e
    str = *context;
  el se
    str = string;
                                                                      string
                                              ptr
                                                            0
  // Find the next nondelimiter
                                                           \ 0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                            W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                            d
        *(str++) = ' \setminus 0';
                                            context
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                            Η
  if( string == NULL )
                                                            e
    str = *context;
  el se
    str = string;
                                                                      string
                                              ptr
                                                            0
  // Find the next nondelimiter
                                                           \ 0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                            W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                        str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                            d
        *(str++) = ' \setminus 0';
                                                           \0
                                            context
       break;
                                                            П
  *context = str;
                                                           \ 0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                           Η
  if( string == NULL )
                                                            e
    str = *context;
  el se
    str = string;
                                              ptr
                                                                      string
                                                            0
  // Find the next nondelimiter
                                                           \ 0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                            W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                            d
        *(str++) = ' \setminus 0';
                                            context
                                                           \0
       break;
                                                            П
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                           Η
  if( string == NULL )
                                                            e
    str = *context;
  el se
    str = string;
                                              ptr
                                                                      string
                                                           0
  // Find the next nondelimiter
                                                           \0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                           0
  for( ; *str != '\0'; str++ )
                                                                       str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                           d
        *(str++) = ' \setminus 0';
                                                           \ 0
                                            context
       break;
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                           Η
  if( string == NULL )
                                                            e
    str = *context;
                                                            1
  el se
                                                            1
    str = string;
                                              ptr
                                                           0
  // Find the next nondelimiter
                                                           \0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                           0
  for( ; *str != '\0'; str++ )
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                           d
        *(str++) = ' \setminus 0';
                                                           \0
       break;
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                            Η
  if( string == NULL )
                                                            e
    str = *context;
                                                            1
  el se
                                                            1
    str = string;
                                                                      string
                                              ptr
                                                            0
  // Find the next nondelimiter
                                                            \ 0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                            W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                        str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                            d
        *(str++) = ' \setminus 0';
                                                            \ 0
                                            context
       break;
  *context = str;
                                                            \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                            Η
  if( string == NULL )
                                                            e
    str = *context;
                                                            1
  el se
                                                            1
    str = string;
                                                                      string
                                              ptr
                                                            0
  // Find the next nondelimiter
                                                            \ 0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                            W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                                        str
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                            d
        *(str++) = ' \setminus 0';
                                                            \ 0
                                            context
       break;
                                                            ш
  *context = str;
                                                            \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                            Η
  if( string == NULL )
                                                            e
    str = *context;
                                                            1
  el se
                                                            1
    str = string;
                                                                      string
                                              ptr
                                                            0
  // Find the next nondelimiter
                                                           \ 0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                            W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                            d
        *(str++) = ' \setminus 0';
                                                           \ 0
                                            context
       break;
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                            Η
  if( string == NULL )
                                                            e
    str = *context;
                                                            1
  el se
                                                            1
    str = string;
                                                                      string
                                              ptr
                                                            0
  // Find the next nondelimiter
                                                           \ 0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                            W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                            d
        *(str++) = ' \setminus 0';
                                            context
                                                           \0
       break;
                                                            ш
  *context = str;
                                                            \ 0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

```
char *strtok_s( char *string, const char *control, char **context )
  char *str:
                                                           Η
  if( string == NULL )
                                                            e
    str = *context;
                                                            1
  el se
                                                            1
    str = string;
                                              ptr
                                                           0
  // Find the next nondelimiter
                                               0
                                                           \0
  while( belong( *str, control ) )
    str++:
  string = str;
                                                           W
                                   nextToken
  // Find the next delimiter
                                                            0
  for( ; *str != '\0'; str++ )
                                                            r
    if( belong( *str, control ) )
                                                            1
    {
                                                           d
        *(str++) = ' \setminus 0';
                                                           \0
       break;
                                                            ш
  *context = str;
                                                           \0
                        ptr = strtok_s( string, "\",! ", &nextToken );
  if( string == str )
                        while( ptr != NULL )
    return NULL;
                        {
  el se
                           cout << ptr << '\n';
    return string;
                           ptr = strtok_s( NULL, "\",! ", &nextToken );
```

/\*

char \*strtok(string, control) - tokenize string with delimiter in control

### Purpose:

strtok considers the string to consist of a sequence of zero or more text tokens separated by spans of one or more control chars. the first call, with string specified, returns a pointer to the first char of the first token, and will write a null char into string immediately following the returned token. subsequent calls with zero forthe first argument (string) will work thru the string until no tokens remain. the control string may be different from call to call. when no tokens remain in string a NULL pointer is returned. remember the control chars with a bit map, one bit per ascii char. the null char is always a control char.

### Entry:

token

char \*string - string to tokenize, or NULL to get next

char \*control - string of characters to use as delimiters

#### Exi t:

returns pointer to first token in string, or if string was NULL, to next token returns NULL when no more tokens remain

```
char *strtok( char *string, const char *control )
{
  char *str;
  static char *context = string;

/* Initialize str */

/* If string is NULL, set str to the saved pointer (i.e., continue breaking tokens out of the string from the last strtok call) */
  if( string == NULL )
    str = context;
  else
    str = string;
```

```
/* Find beginning of token (skip over leading delimiters).
  Note that there is no token iff this loop sets str to point
   to the terminal null (*str == ' \0') */
while( belong( *str, control ) )
  str++:
string = str;
/* Find the end of the token. If it is not the end of the string,
   put a null there. */
for( ; *str != '\0'; str++ )
  if( belong( *str, control ) ) {
     *(str++) = ' \setminus 0';
     break:
/* Update nextoken */
context = str;
/* Determine if a token has been found. */
if( string == str )
  return NULL;
el se
  return string;
```

# 國立中正大學103學年度碩士班招生考試試題

3. (6%) The following C function converts a string of lowercase characters into uppercase characters. forexample, it will convert "abc" into "ABC". The parameter **str** is the string. Fill in the missing expressions in (1) and (2).

```
void lower2upper(char str []) {
   int i = 0;
   while(__(1)__) {
      str[i] = str[i] + __(2)__;
      i++;
   }
}
```

# 國立中正大學103學年度碩士班招生考試試題

3. (6%) The following C function converts a string of lowercase characters into uppercase characters. forexample, it will convert "abc" into "ABC". The parameter **str** is the string. Fill in the missing expressions in (1) and (2).

```
void lower2upper(char str []) {
   int i = 0;
   while(_str[i] != '\0'_) {
      str[i] = str[i] + _'A' - 'a'_;
      i++;
   }
}
```